1) (25 points) Valider une date

Vous travaillez pour une compagnie manufacturière de meubles et votre employeur met sur pied un système de gestion de commandes et de livraison en ligne. À l'aide de ce système, vos clients auront la possibilité d'entrer, eux-mêmes, la date de livraison de leurs commandes.

Définissez des constantes (3 Points) afin de représenter chaque mois de l'année.

Écrivez une fonction (5 Points) qui reçoit un nombre et les bornes supérieures et inférieures d'un intervalle. Cette fonction retourne « True » si le nombre reçu est à l'intérieur de l'intervalle reçu et « False » lorsque ce n'est pas le cas.

Exemple: Si les bornes sont entre [0; 10] et que le nombre entré est 10, la fonction retourne «True».

Exemple: Si les bornes sont entre [0; 10] et que le nombre entré est -8, la fonction retourne «False».

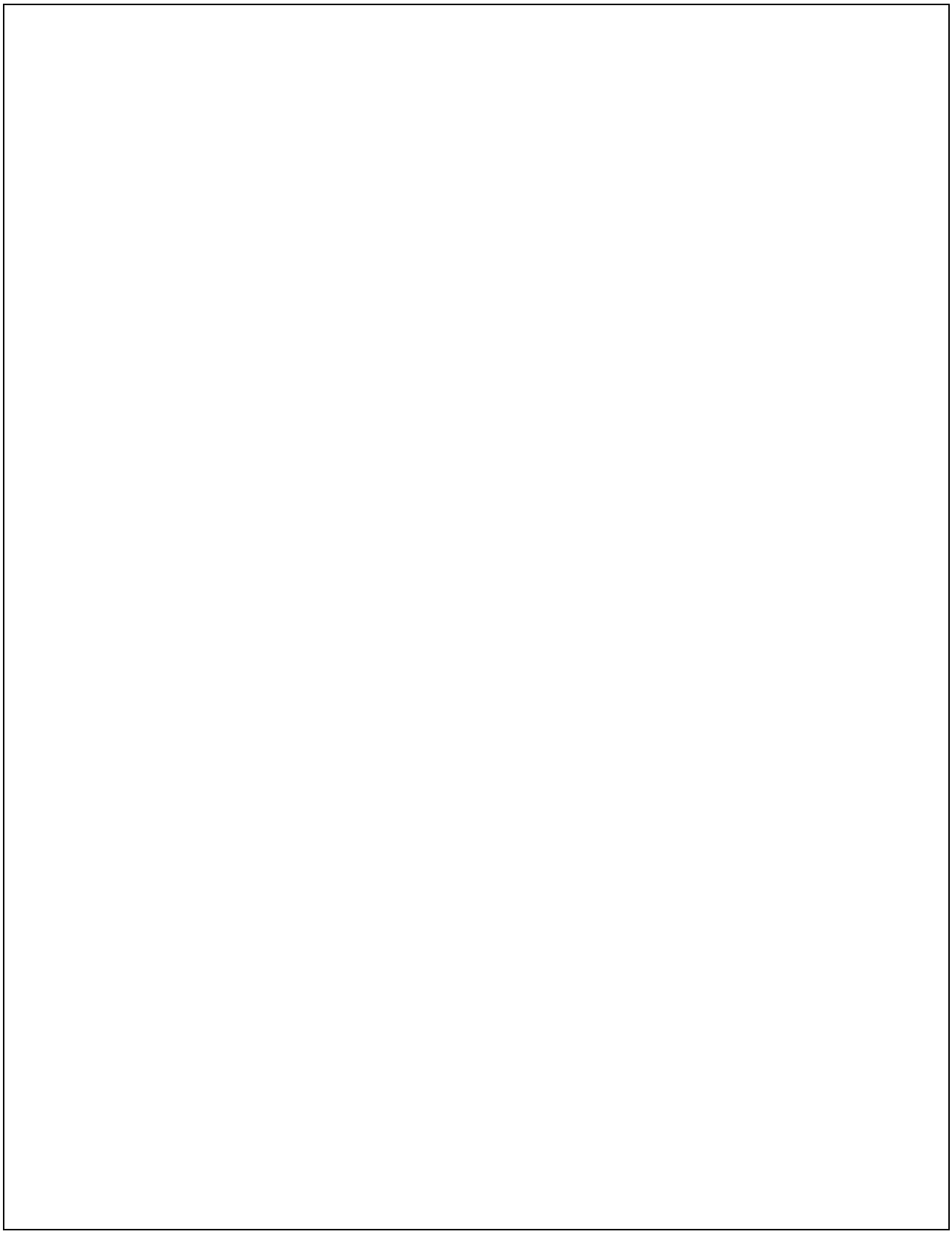
Puisque vous offrez des livraisons 7 jours sur 7, même pendant les journées fériées, votre système doit seulement s'assurer que la date entrée existe. Afin de gérer les années bissextiles, un de vos collègues vous fournit une fonction dont voici l'entête.

```
%************************************************************************
% ANNEE_EST_BISSEXTILE
%
% Cette fonction indique si l'année reçue en paramètre est une année
% bissextile.
%
% Paramètres:
%     - [Integer] année: L'année à valider.
%
% Retour :
%     - [Boolean] True lorsque année est bissextile, False sinon.
%************************************************************************
function [est_bissextile] = annee_est_bissextile(annee)
```

Écrivez une fonction nommée date_est_valide (17 Points) qui reçoit un jour, un mois et une année. La fonction retourne « True » si la date reçue est valide. Voici les vérifications que vous devez réaliser :

- Une année bissextile possède 29 jours dans le mois de février.
- Une année non bissextile possède 28 jours dans le mois de février.
- Les mois de janvier, mars, mai, juillet, août, octobre et décembre possèdent 31 jours.
- Les mois d'avril, juin, septembre et novembre possèdent 30 jours.

(Continuer à la page suivante si vous n'avez pas assez de place)



2) (25 points) Plus haut ou plus bas?

Afin d'initier vos jeunes enfants aux ordinateurs, vous décidez de programmer un jeu de dés. Voici comment se déroule le jeu. Tout d'abord, le jeu débute par un lancer de dé initial. Vous devez afficher le résultat du lancer au joueur et lui demander de deviner correctement si le prochain dé lancé sera :

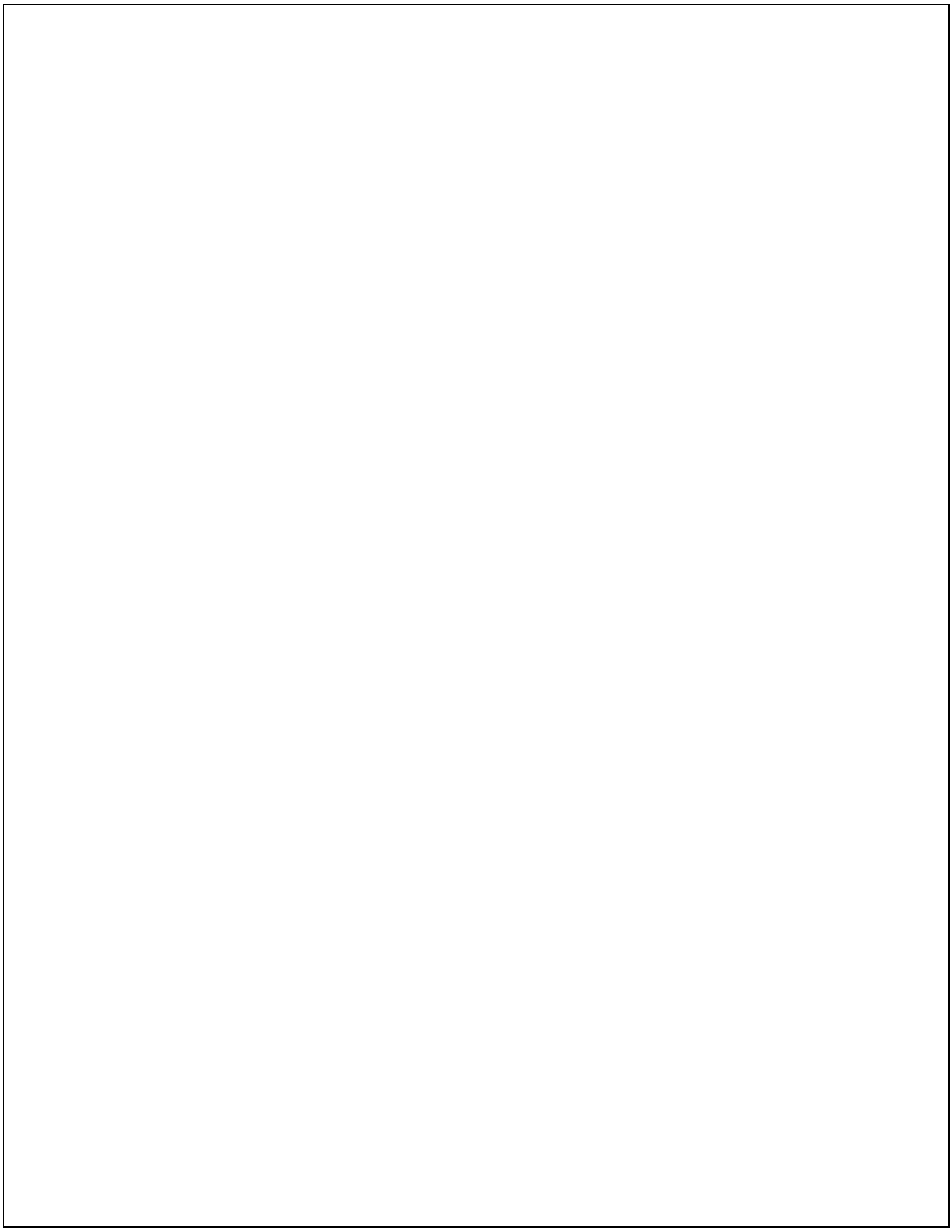
- Plus grand que le lancer précédant.
- Plus petit que le lancer précédant.
- Égal au lancer précédant.

Une fois le résultat saisi, le programme lance à nouveau un dé et vérifie si la prédiction était correcte. Ensuite, vous indiquerez à l'utilisateur si sa prédiction était correcte ou erronée. On recommence ce processus tant et aussi longtemps que les prédictions du joueur sont correctes. Chaque bonne prédiction lui vaut un point. Lorsque le joueur fait une mauvaise prédiction, le jeu se termine et le pointage du joueur est affiché.

Il est possible d'obtenir le résultat d'un lancer de dé, à l'aide de la fonction suivante :

```
%**************************************************************************
% LANCER_UN_DE
%
% En considérant un dé ayant des faces numérotées entre face_minimale et
% face_maximale; et en considérant que chaque face est présente une seule
% fois sur le dé; cette fonction retourne le résultat d'un lancer de dé.
%
% Paramètres:
%     - [Integer] face_minimale: La plus petite face du dé.
%     - [Integer] face_maximale: La plus grande face du dé.
%
% Retour :
%     - [Integer] Une valeur comprise entre face_minimale et face_maximale
%                 représentant un lancé de dé.
%**************************************************************************
function [de] = lancer_un_de(face_minimale, face_maximale)
```

Écrivez une procédure qui permet de jouer au jeu décrit. **À noter que l'exécution fournie à la page suivante est un exemple**. Les valeurs des dés ont été obtenues aléatoirement à l'aide de la fonction `lancer_un_de`. Vous devez absolument utiliser cette fonction. Le nombre de faces de sur votre dé **doit être identifié par une constante.**



Voici un exemple d'exécution du programme pour un dé à 20 faces : Le résultat du premier lancé de dé était de 13, voici ce qui doit être affiché à l'utilisateur :

```
La valeur du dernier lancé de dé était de 13
Le prochain lancé de dé sera :
1 - Plus haut
2 - Plus bas
3 - Égal
```

Le joueur prédit que le prochain dé sera plus bas en entrant 2. On lance le dé et on obtient un résultat de 6. On affiche à l'utilisateur ce résultat et on le félicite pour sa bonne prédiction.

```
Bravo! La valeur du dé est de : 6
```

Puis, on lui demande de prédire à nouveau le résultat du prochain dé.

```
La valeur du dernier lancé de dé était de 6
Le prochain lancé de dé sera :
1 - Plus haut
2 - Plus bas
3 - Égal
```

Il décide de prédire un résultat plus haut, mais malheureusement le résultat du prochain lancer est de 4. On affiche à l'utilisateur ce résultat, on lui indique qu'il s'est trompé et on affiche son pointage final.

```
Malheureusement, la valeur du dé est de : 4
Vous avez obtenus un score de : 1
```

3) (15 points) Racine nième

Écrivez une fonction qui reçoit un nombre ainsi que la valeur n. Cette fonction retourne la racine n^{ième} d'un nombre. La racine n^{ième} d'un nombre peut être calculée à l'aide des trois étapes suivantes.

a) on fixe racine à nombre.

b) on trouve la prochaine valeur de racine à l'aide de l'équation suivante :

$$racine = \frac{1}{n}\left[(n-1) * racine + \frac{nombre}{racine^{n-1}}\right]$$

c) on répète b) jusqu'à ce que racine soit suffisamment près de nombre
(i.e. |racine^{n} - nombre|≈0)

On vous demande une précision de 0.001 sur la racine calculée. Utilisez la fonction abs( ) afin de calculer la valeur absolue. Déclarez vos constantes.